

Apuntadores y Manejo de archivos en C

L.I. Francisco Ruiz Sala
Instituto de Astronomía
Facultad de Ciencias
UNAM

¿Que es un apuntador?

Llamados también punteros.

“Un Apuntador es una variable que contiene una dirección de memoria, la cual corresponderá a un dato o a una variable que contiene el dato.

Cada variable que se utiliza en una aplicación ocupa una o varias posiciones de memoria.

Estas posiciones de memoria se accesan por medio de una dirección”

Sintaxis

La declaración de un puntero de manera general es:

Tipo_dato *nombre de apuntador;

Tipo_dato : Especifica el tipo de objeto apuntado y puede ser cualquier tipo (int, float, char, etc).

Nombre de apuntador: Es el identificador (nombre asignado) del apuntador.

Ejemplos de declaración:

```
int *ptr, cont; float *res; short *bandera; char *mensaje;
```

Ejemplo:

```
#include <stdio.h>
int main()
{
    char a;          /* Variable 'a' de tipo char */
    char *pchar;     /* Puntero a char 'pchar' */
    a=65;            //inicializamos a

    pchar = &a;      /* 'pchar' <- @ de 'a' */

    printf("la direccion de memoria de 'a' es: %p \n", &a);
    printf("y su contenido es : %c \n", *pchar);

    // Verificación de tabla ascii https://elcodigoascii.com.ar/
}
```

¿Que es un archivo?

Es un conjunto de bytes que son almacenados en un dispositivo (electronico, electromecanico).

El mismo es identificado por un nombre y características que contiene, dependiendo el Sistema Operativo y el tipo de particion en la cual se almacene.

A los archivos informáticos se les llama así porque son los equivalentes digitales de los archivos escritos en expedientes, tarjetas, libretas, papel o microfichas del entorno de oficina tradicional.

Tipos de Archivos:

- Los Archivos en Unix dependen de los permisos
- Tipo texto, Binarios, Otros.
- Otros: Directorios, Ligas, ocultos, shell, de sistema.
- `rwXrw----`

Manejo de Archivos en C

- Al depender de Unix, los archivos pueden ser de texto claro o binarios
- Las interfaces de entrada y salida.
- Interfaz estándar de Entrada Teclado, Salida Monitor.
- El archivo, como cualquier dispositivo es de entrada y salida

Qué es un flujo o stream:

- El conjunto de datos que transfieres desde un programa a un archivo, o al revés, se le conoce como *flujo* (*stream* en inglés), y consiste en una serie de bytes (o caracteres). A diferencia de un archivo, que se refiere a un dispositivo concreto de E/S, un flujo es independiente del dispositivo.
- Es decir, todos los flujos tienen el mismo comportamiento independientemente del dispositivo al que esté asociado.
- De esta manera puedes realizar operaciones E/S con tan sólo asociar un flujo a un archivo.
- Hay dos tipos de flujos:
 - Uno es el *flujo de texto*, consistente en líneas de texto. una línea es una secuencia de caracteres terminada en el carácter de línea nueva ('\n' ó '\r\n').
 - El otro formato es el del *flujo binario*, que consiste en una secuencia de bytes que representan datos internos como números, estructuras o arrays. se utiliza principalmente para datos que no son de tipo texto, donde no importa la apariencia de esos datos en el archivo (da igual que no se vean "en bonito", como podría verse en un archivo de texto).

E/S mediante buffers

- Una porción de memoria que se usa temporalmente para almacenar datos antes de ser enviados a su destino se llama buffer. Con ayuda de los buffers, el sistema operativo puede mejorar su eficiencia reduciendo el número de accesos a efectuar en un dispositivo de E/S (es decir, en un archivo).
- El acceso a disco o a otros dispositivos de E/S suele ser más lento que a memoria de acceso directo; por eso, si varias operaciones E/S se realizan en un buffer en vez de en el archivo en sí, el funcionamiento de un programa es más eficiente. Por ejemplo, si se envían varias operaciones de escritura a un buffer, las modificaciones de esos datos escritos se quedan en memoria hasta que es hora de guardarlas, que es cuando esos datos se llevan al dispositivo real (a este proceso se le conoce como flushing).
- Por defecto, todas las operaciones de los flujos E/S en C son con buffer.

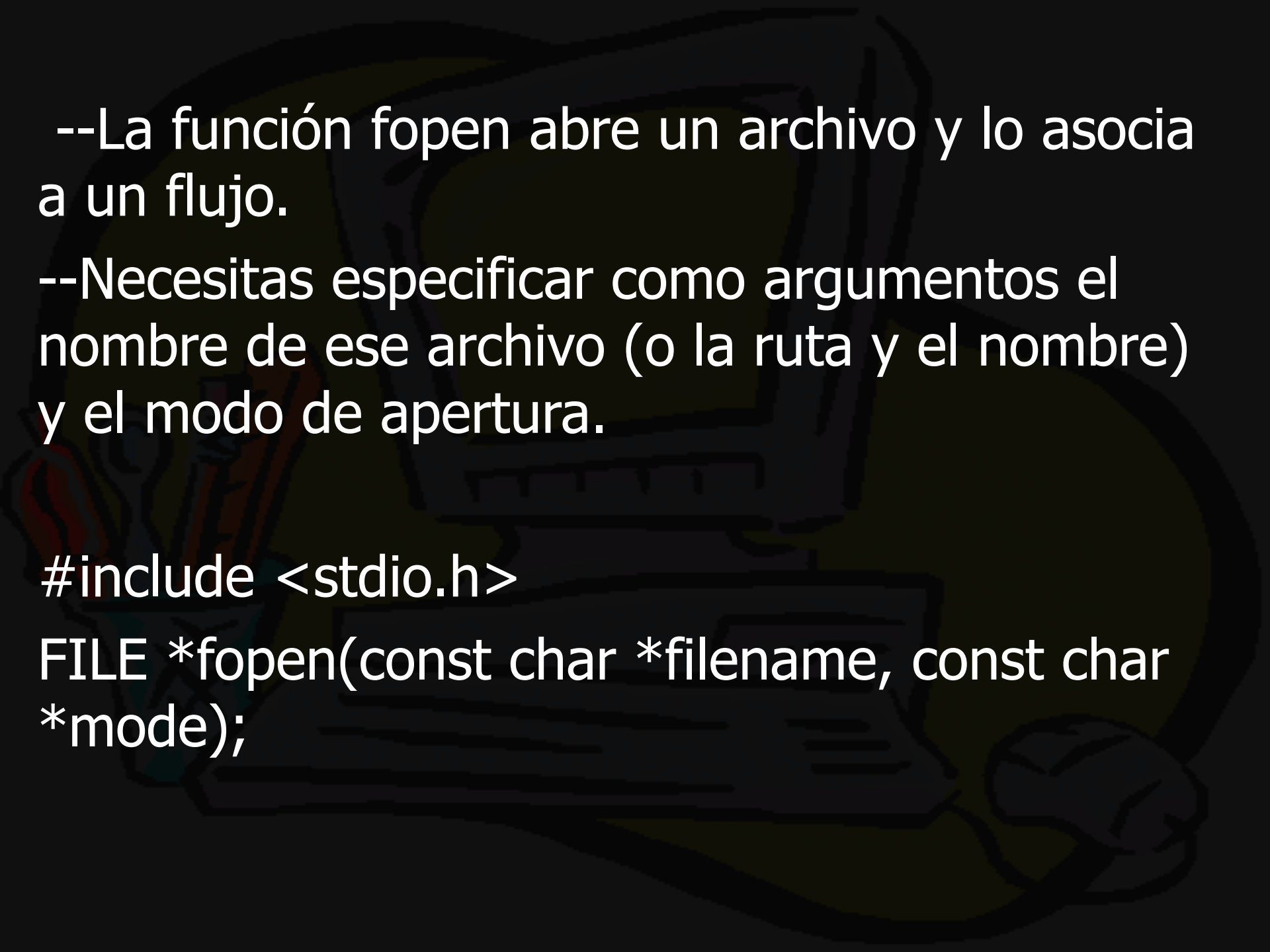
Comportamiento del modelo E/S:

- Ya que la unidad más pequeña que se puede representar en C es un carácter (char), se puede acceder a un archivo desde cualquiera de sus caracteres (o bytes, que es lo mismo).
- Se puede leer o escribir cualquier número de caracteres desde un punto móvil, conocido como el indicador de posición de archivo.
- Los caracteres se leen o escriben en secuencia desde ese punto, y el indicador se va moviendo de acuerdo a eso.
- El indicador se coloca al principio del archivo cuando éste se abre, pero luego se puede mover a cualquier punto explícitamente (no sólo porque se haya leído o escrito algo).

Punteros a FILE:

- La estructura FILE es la estructura que controla los archivos y se encuentra en la cabecera stdio.h.
- Los flujos utilizan estos punteros a archivos para realizar las operaciones E/S.
- La siguiente línea de código declara una variable de tipo puntero a archivo:

```
FILE *ptr_fich;
```



--La función fopen abre un archivo y lo asocia a un flujo.

--Necesitas especificar como argumentos el nombre de ese archivo (o la ruta y el nombre) y el modo de apertura.

```
#include <stdio.h>
```

```
FILE *fopen(const char *filename, const char  
*mode);
```

--Aquí filename es un puntero a char que referencia un string con el nombre del archivo.

-- mode apunta a otro string que especifica la manera en la que se va a abrir el archivo. La función fopen devuelve un puntero de tipo FILE. Si ha ocurrido algún error al abrirse, devuelve NULL.

--El parámetro mode es una combinación de los caracteres r (read, lectura), w (write, escritura), b (binario), a (append, añadir), y + (actualizar).

--Si usas el caracter a y el archivo existe, el contenido de ese archivo se mantiene y los datos nuevos se añaden justo al final, tras el último dato que ya estuviera escrito (el indicador de archivo pues no está situado al inicio del archivo, sino que en este caso está al final).

--Si el archivo no existe, lo crea. Con w es diferente; este modo siempre borra los datos del archivo si existe (si no, crea uno nuevo).

--Si al modo le añades el +, permite que el archivo se abra para escritura o lectura y puedes modificar cualquier dato que hubiera en él. Si usas r, el archivo debe existir; si no, fopen dará error y te devolverá NULL.

La siguiente lista muestra los posibles modos de abrir un archivo:

"r" abre un archivo de texto existente para lectura.

"w" crea un archivo de texto para escritura.

"a" abre un archivo de texto existente para añadir datos.

"r+" abre un archivo de texto existente para lectura o escritura.

"w+" crea un archivo de texto para lectura o escritura.

"a+" abre o crea un archivo de texto para añadirle datos.

"rb" abre un archivo binario existente para lectura.

"wb" crea un archivo binario para escritura.

"ab" abre un archivo binario existente para añadir datos.

"r+b" abre un archivo binario existente para lectura o escritura.

"w+b" crea un archivo binario para lectura o escritura.

"a+b" abre o crea un archivo binario para añadirle datos.

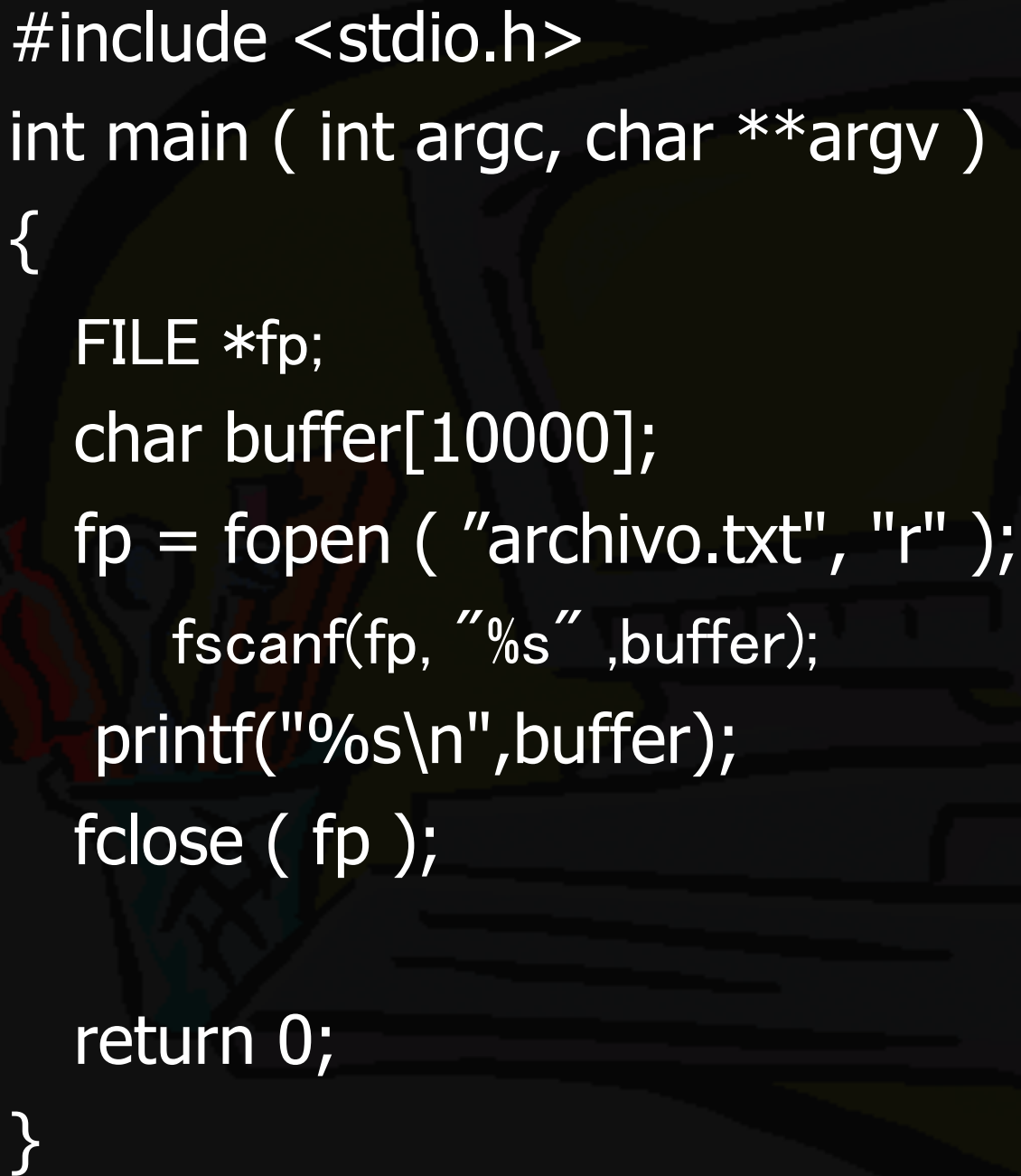
A veces podrás ver que el modo está escrito como "rb+" en vez de "r+b", por ejemplo, pero es equivalente.

Cerrar un Archivo:

Después de abrir un archivo y leerlo o escribir en él, hay que desenlazarlo del flujo de datos al que fue asociado. Esto se hace con la función `fclose`, cuya sintaxis es la siguiente:

```
#include <stdio.h>  
int fclose(FILE *stream);
```

Si `fclose` cierra bien el archivo, devuelve 0. Si no, la función devuelve EOF. Normalmente sólo falla si se ha borrado el archivo antes de intentar cerrarlo. Acuérdate siempre de cerrar el archivo. Si no lo haces, se pueden perder los datos almacenados en él. Además, si no lo cierras, otros programas pueden tener problemas si lo quieren abrir más tarde.



```
#include <stdio.h>

int main ( int argc, char **argv )
{
    FILE *fp;
    char buffer[10000];
    fp = fopen ( "archivo.txt", "r" );
        fscanf(fp, "%s" ,buffer);
    printf("%s\n",buffer);
    fclose ( fp );

    return 0;
}
```




```
#include <stdio.h>
```

```
enum {EXITO, FALLO};
```

```
int main(void)
```

```
{
```

```
    FILE *ptr_fichero;
```

```
    char nombre_fichero[] = "resumen.txt";
```

```
    int resultado = EXITO;
```

```
    if ( (ptr_fichero = fopen(nombre_fichero, "r") ) == NULL)
```

```
    {
```

```
        printf("No se ha podido abrir el fichero %s.\n", nombre_fichero);
```

```
        resultado = FALLO;
```

```
    }
```

```
    else
```

```
    {
```

```
        printf("Abierto fichero; listos para cerrarlo.\n");
```

```
        if (fclose(ptr_fichero)!=0)
```

```
        {
```

```
            printf("No se ha podido cerrar el fichero %s.\n", nombre_fichero);
```

```
            resultado = FALLO;
```

```
        }
```

```
    }
```

```
    return resultado;
```

```
}
```



```
#include <stdio.h>
```

```
#include <stdlib.h>
```

```
int main()
```

```
{ FILE *archivo;
```

```
char letra;
```

```
char cadena[1024];
```

```
archivo = fopen("origen.txt","r");
```

```
if (archivo == NULL) {
```

```
    printf( "No se puede abrir el archivo.\n" );
```

```
    exit( 1 );
```

```
}
```

```
printf( "Contenido del archivo:\n" );
```

```
letra = getc(archivo);
```

```
while (feof(archivo) == 0) {
```

```
    printf( "%c",letra );
```

```
    letra = getc(archivo);
```

```
}
```


```
if (fclose(archivo)!= 0)
```

```
    printf( "Problemas al cerrar el archivo\n" );
```

```
}
```

```
#include <stdio.h>
#include <stdlib.h>
struct threeNum
{
    int n1, n2, n3;
};
int main()
{
    int n;
    struct threeNum num;
    FILE *fptr;
    if ((fptr = fopen("prueba.bin", "wb")) == NULL){
        printf("Error! opening file");
        // Program exits if the file pointer returns NULL.
        exit(1);
    }
    for(n = 1; n < 5; ++n)
    {
        num.n1 = n;
        num.n2 = 5*n;
        num.n3 = 5*n + 1;
        fwrite(&num, sizeof(struct threeNum), 1, fptr);
    }
    fclose(fptr);

    return 0;
}
```



```
#include <stdio.h>
#include <stdlib.h>

struct threeNum { int n1, n2, n3;};

int main()
{
    int n;
    struct threeNum num;
    FILE *fptr;
    if ((fptr = fopen("prueba.bin","rb")) == NULL){
        printf("Error! opening file");
        // Si el archive existe el punter es NULO o NULL.
        exit(1);
    }
    for(n = 1; n < 5; ++n)
    {
        fread(&num, sizeof(struct threeNum), 1, fptr);
        printf("n1: %d\t n2: %d\t n3: %d\n", num.n1, num.n2, num.n3);
    }
    fclose(fptr);
    return 0;
}
```

// <https://www.programiz.com/c-programming/c-file-input-output>